Data sharing in DHT based P2P systems

Claudia Roncancio[1], María del Pilar Villamil[2], Cyril Labbé[1], and Patricia Serrano-Alvarado[3]

[1]University of Grenoble, France**
firstName.lastName@imag.fr
[2]University of Los Andes, Bogotá, Colombia,
mavillam@uniandes.edu.co
[3]University of Nantes, France
Patricia.Serrano-Alvarado@univ-nantes.fr

Abstract. The evolution of peer-to-peer (P2P) systems triggered the building of large scale distributed applications. The main application domain is data sharing across a very large number of highly autonomous participants. Building such data sharing systems is particularly challenging because of the "extreme" characteristics of P2P infrastructures: massive distribution, high churn rate, no global control, potentially untrusted participants... This article focuses on declarative querying support, query optimization and data privacy on a major class of P2P systems, that based on Distributed Hash Table (P2P DHT). The usual approaches and the algorithms used by classic distributed systems and databases for providing data privacy and querying services are not well suited to P2P DHT systems. A considerable amount of work was required to adapt them for the new challenges such systems present. This paper describes the most important solutions found. It also identifies important future research trends in data management in P2P DHT systems.

Key words: DHT, P2P Systems, Data sharing, Querying in P2P systems, Data privacy

1 Introduction

Peer-to-peer (P2P) systems take advantage of advances in networking and communication for providing environments where heterogeneous peers with high autonomy compose a system with a fully distributed control. P2P systems are the chosen platform for new style of applications where distributed data can be shared massively e.g., social networks [62], geo-collaboration systems [42], professional communities (medical, research, open-source software [2]).

The development of massively distributed data sharing systems raises new and challenging issues. This results from the intrinsic characteristics of P2P systems (distribution among a huge number of peers, dynamic systems configuration, heterogeneity of data and peers, autonomy of data sources, very large

** This work is supported by the ECOS C07M02 action.

volume of shared data) which prevent the direct use of distributed algorithms issued from the more classical distributed systems and database worlds. Consequently, new approaches are being proposed but also existing algorithms are being revisited and adapted to provide high level data management facilities in the P2P context.

The P2P context includes a large variety of systems, ranging from overlay networks and distributed lookup services, until high level data management services. Structured and unstructured overlay P2P networks exist and lead to systems with different characteristics.

This paper concerns data sharing in structured P2P systems based on a Distributed Hash Table (DHT). It provides a synthesis of the main proposals on efficient data querying and data privacy supports. These two aspects are essential but challenging to implement in massively distributed data sharing systems.

The absence of a global view and control of a system composed by a large set of volatile participants which can be both, data providers and requesters, implies that new querying mechanisms are needed. Query processors rely on the underlying overlay network and are expected to follow the P2P approach without introducing centralization points. This paper gives an overview of the main querying solutions proposed for systems sharing semi-structured or relational data but also for data type independent systems where queries are based on meta-data (attributes, keywords, etc). It also briefly discusses rich information retrieval approaches. Query languages, index structures and other optimization solutions, such as caches, will be analyzed.

As P2P systems are very attractive to some communities (e.g., professional ones) willing to share sensitive or confidential data in a controlled way, data privacy support is an important issue. Nevertheless, the open and autonomous nature of P2P systems makes hard to provide privacy of peers and data. The P2P environment can be considered as hostile because peers are potentially untrusted. Data can be accessed by everyone and anyone, used for everything and anything[1] and a peer's behavior and identity can be easily revealed. This paper discusses recent results on privacy support on P2P DHT systems. It analyzes proposals to ensure a peer's anonymity, to improve data access control and to use/adapt trust techniques.

The paper is organized as follows. Section 2 introduces P2P DHT systems and a functional architecture of them. Section 3 analyzes several representative proposals providing declarative queries on top of DHT systems. It discusses the main design choices concerning data, meta-data, language expressiveness and index. More specific optimization aspects are discussed in Section 4. Section 5 concentrates on privacy issues. Section 6 concludes this paper and gives research perspectives on P2P DHT data management.

[1] Profiling, illegal competition, cheating, marketing or simply for activities against the owner's preferences or ethics.

2 System support for P2P DHT systems

This section introduces a functional architecture for P2P DHT systems and its main provided services. It focuses on the specific services used by the high level querying solutions that will be analyzed in Section 3.

P2P DHT systems have known great success mainly because their high scalability to organize large sets of volatile peers. They use a self-organizing overlay network on top of the physical network. One of their key characteristics is the efficient lookup service they provide and, very important in data sharing, their excellent support to provide comprehensive answers to queries.

Several proposals exist [63, 73, 66, 49]. They rely on the use of a distributed hash table to index participant peers and shared objects by using hash keys as identifiers. A DHT system splits a key space into zones and assigns each zone to a peer. A key is a point in this space and the object corresponding to this key is stored at the peer whose zone contains this point. Locating an object is reduced to routing to the peers hosting the object. One of the main differences between DHT solutions is their routing geometry. For instance, CAN [63] routes along a d-dimensional Cartesian space, Chord [73] and Pastry [66] along a ring and Viceroy [49] uses a butterfly network. Techniques used in the routing process are the fundamentals to build P2P DHT systems but several other services have been proposed to build more complete infrastructures. One of the reasons is semantic less access provided at this level: to find an object in such a system, its key has to be known. This key is then efficiently located among the peers in the system. Such a query, called **location query**, retrieves the physical identifier of the peer where the object is stored.

Several proposals extending P2P DHT systems exist but there is no standard yet [21]. Let's consider the three layer functional architecture presented in Figure 1. The lower level, labeled **Distributed Lookup Service**, provides the overlay network support, the second layer, **Distributed Storage Service**, provides data persistence services and the third layer, **Distributed Data Services**, manages data as semantic resources.

The **Distributed Lookup Service** provides efficient mechanisms to find peers (identified by keys) on a distributed system by managing the routing information (neighbors set of a peer). Their main functions are:

- *lookup(key):* returns the physical identifier of the peer in charge of a key.
- *join(key):* registers a new peer in the distributed system and updates routing information.
- *leave(key):* handles the departure of a peer and updates routing information.
- *neighbors(key):* returns a list of keys identifying the peers in charge of the neighbor zones.

This basic layer is used to build systems that give semantics to keys and provide data storage.

Distributed Storage Services are responsible for stored object administration – insertion, migration, etc. Objects migrate from a peer that leaves the system to one of its neighbors to insure the object's durability. Object migration

and replication (in neighbors of their storage peers) work to ensure that stored values never disappear from the system. Load balancing and caching strategies are also proposed. Examples of systems of this level are PAST [67], CFS [20] and DHash [29].

The main external functions of this layer are:

- *get(key):* returns the object (or collection) identified by *key*. It uses the *lookup* function to locate the peers storing such objects. Answers are comprehensive, i.e., all answers present in the system are retrieved[2].
- *put(key, Object):* inserts into the system an object identified by its key. It uses the *lookup* function to identify the peer where the object has to be stored.

It is worth remarking that an object stored in such systems becomes a shared object. In practice, it is not easy to remove objects as temporally absent peers may have a copy of a deleted object. Coherency problems may arise when such peers are restored.

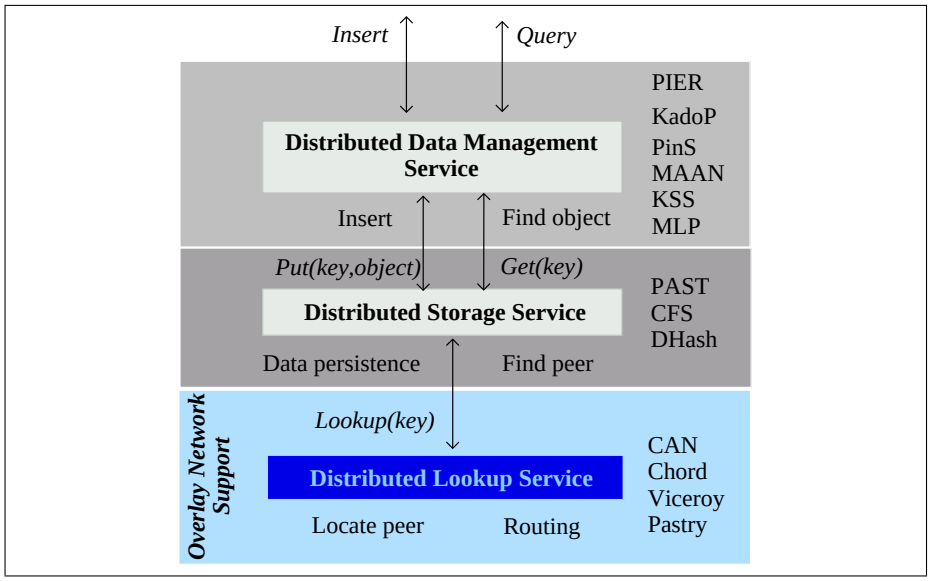

Fig. 1. Functional architecture of P2P DHT systems

The higher level of the proposed functional architecture offers services to allow semantic management of shared objects. This layer is important because underlying layers do not provide keyword search functions, multi attribute searches neither comparative query evaluation. They only offer access to objects giving their key [3] and more powerful queries are hard to handle since the objects dis-

[2] This characteristic can not be easily provided by unstructured P2P systems.
[3] Obtained through a known hash function.

tribution criteria is based on semantic less keys. Several important proposals to improve querying capabilities have been proposed [33, 26, 29, 15, 74, 74, 30, 75, 4]. They rely on distributed lookup and storage services. Chord is adopted by KSS [29], MAAN [15] and FCQ[74]. PAST has been chosen by PinS [75] and Kadop [4] whereas CAN is exploited by PIER [33]. Such high level querying services are presented in Section 3.

To finish this section, it is worth mentioning some efforts [37, 11, 19] to optimize specific aspects of distributed lookup services. Proposals such as Baton [37] and Mercury [11] modify the structure of the overlay network to improve some kinds of research on keys. They concern mainly the optimization of range queries or queries concerning intervals of keys. Baton proposes a balanced tree structure to index peers whereas Mercury organizes peers in groups. Each group of peers takes in charge the index of an attribute. Peers in a group are organized in a ring and the attribute domain is divided to give the responsibility of an interval of values to each peer. Mercury proposes an integrated solution without distinction of the three layers.

3 Declarative queries

The purpose of this section is to present an overview of a representative sample of works dealing with the evaluation of declarative high level queries in P2P DHT systems. As seen in the previous section, basic configuration of P2P DHT systems remains limited in term of research. Multi-attribute search based on equality and inequality criteria, or including join and aggregates operations, are not possible. Breaking this lack of semantic is a crucial point for the future of P2P systems. Significant progress have been achieved thanks to a lot of work made to improve declarative querying support.

3.1 Global view

The evaluation of declarative queries is made at the data management layer which relies on underlying layers (storage management and overlay network). Services provided by these layers are very important because they limit or allow some flexibility to data management services.

Querying facilities offered by data management services are closely related to the nature of shared data and used meta-data. One can find dedicated services to a single type of data – for example [28] – or rather generic systems allowing sharing various types of resources such as [70]. The choice of meta-data used to describe and index resources determines usable criteria to formulate queries. One can find approaches that use keywords [29, 64, 70] whereas others are based on attributes [34, 27, 76].

In most cases [15, 29, 74, 34, 27], meta-data are stored using the storage management layer. Data and meta-data are identified by a key obtained with a hash function. Such an identifier will be associated, with a keyword [70, 29], an attribute [29, 28], a couple (attribute, value) [76], or a path in XML [27].

The equality operator is proposed by a wide range of systems [29, 70]. More complex operators, like inequality, are much more difficult to implement because of the "blink" storage of shared resources. The storage management layer deals with the placement of shared resources by using hash functions to decide which peer will ensure the storage. This approach, which does not consider any semantics, makes it impossible to determine the most useful peer to help to resolve the query. One can find in [27, 33, 76], interesting solutions for query containing inequality operators. Even more complex operators like joining or sorting can be found in [74, 34, 60].

Massively distributed query processing systems came with new problems including query optimization. With regard to improvement of response time, proposals are based on index, caches [28, 75], duplication [27] and materialized queries [29, 75].

Evolving in a very high scaled environment also leads to Information Retrieval (IR) techniques. As a huge number of objects are shared returning the set of all objects satisfying a query may not be relevant, that is why research trends dealing with top-k operators are important. With high scale also comes a high semantic heterogeneity in data description, finding ways to bridge those semantics is crucial.

In the following we give an overview of four classes of systems that are representative of the data management layer. The first three classes of systems are classified according to the structure adopted for the data and/or the meta-data whereas the fourth one is more related to IR trends.

- Systems using meta-data composed by keywords or <attribut, value> couples: KSS [29], MLP [70], PinS [77, 75] and MAAN [15].
- Systems dealing with the relational model: PIER [33], and FCQ [74].
- Systems using the semi-structured or navigational model: DIP [28], KadoP [4], LDS [27].
- IR approaches: SPRITE [46], DHTop [8] and works presented in [81].

3.2 Attribute based model

This first class of systems adopts simple meta-data composed by <attribut, value> couples. Queries expressed on these attributes allow semantic search of data.

KSS and MLP propose query languages with equality conditions on values of attributes. Whereas in PinS and MAAN queries including equality and inequality conditions are allowed. In addition to indexing techniques other optimization techniques are also targeted.

A Keyword-Set Search System for Peer-to-Peer Networks (KSS) relies on DHash and Chord. The storage of objects and related meta-data is delegated to the storage management layer. Shared data have an identifier and can be described using meta-data that can be either keywords or <attribute, value> couples. Meta-data are used, one by one, to index related object in the DHT. Basically, one

can find in the DHT entries like: <hash(keyword1), {ObjId$_1$, ..., ObjId$_n$ }> or <hash(attribute1,value1), {ObjId$_i$, ..., ObjId$_m$ }>. These entries give respectively all objects, registered in the systems, which satisfy $keyword1$ or have $attribute1 = value1$. Queries are conjunctions of meta-data and answers are comprehensive.

KSS is one of the first works allowing evaluation of equality queries over P2P DHT systems. It also proposes a systematic query materialization strategy that enhances query processing to the detriment of storage usage.

Multi-level Partitioning (MLP) is also a keyword indexing system but it is build over the P2P SkipNet [30] system. Peers are structured in a N_{max} leveled hierarchy of groups. Communication between groups is done using a broadcast process.

This structuration allows a parallel evaluation of queries and the ability to return partial answer. A query is broadcasted to all groups on the next level of the hierarchy. At each level, several groups continue the broadcast of the query. The propagation of the query stops at the last level. Answers from each group are a partial answer that can be returned on the fly to the original peer that sent the query.

Data are stored in the peer to which they belong and are not stored in the storage management layer. Only Meta-data (keywords) are used to index data thanks to the storage management layer. So data are published (indexed) but not stored in the DHT.

MLP is a specific solution relying on the particularities of the underlying lookup service SkipNet. As a matter of fact, the use of the hierarchical structure and broadcasting are bases for query evaluation. This allows intensive use of parallelism during query evaluation. As a drawback, the number of contacted peers increases compared to a solution that does not rely on broadcasting. The fact that data are not stored in the DHT can not assure the comprehensiveness of the answer. Data may have gone with a peer but may be still indexed in the system.

Peer to Peer interrogation and indexing (PinS) relies on PAST and Pastry. Queries can be composed of conjunction and disjunction of equality, inequality criteria and join-like operators [60]. Data can be either private or public. Data which are stored and indexed in the DHT are public and data which are indexed by the DHT but stored in a non-shared storage systems (not in the DHT) are called private.

Several kinds of data indexing strategies in the DHT are proposed. The first one allows equality search and is similar to KSS. Three others are proposed to deal with queries composed of equality, inequality, and/or join terms. Those strategies are based on distributed indexes giving values related to one attribute.

For a single query several evaluation strategy may exists and they can be selected according to the execution context to enhance query processing (see Section 4). PinS uses traditional DHT functionalities. For some of those indexes,

the problem of peer saturation may arise due to the size of the index. To overcome this, PinS proposes dynamic index fragmentation and distribution.

Multi-Attribute Addressable Network (MAAN) adopts the P2P approach for the discovery of shared resources in grids. Like PinS, queries are composed of equality and inequality criteria using meta-data. Shared resources are described using systems oriented meta-data (e.g. name, operating system, CPU). Like others solutions, the query evaluation is based on distributed index. This system uses a hashing function with a special property, namely a *uniform locality preserving hashing function*. This hashing function preserves order and assures a uniform distribution.

MAAN allows the evaluation of inequality terms with a strategy based on its ordered hashing function. There are several drawbacks for solution based on an ordered hashing function. Building the hashing function implies the knowledge of each attributes distribution. For attributes with a large domain of values in which only few are used, the evaluation process will require useless processing in many peers. The evaluation process relies on the successor function which is a functionality that may not be a standard function for P2P systems. This function may not be so simple in some overlay networks.

3.3 Relational model

The second group is composed of works which consider the P2P system as a distributed relational database (PIER, DHTop and FCQ). They allow the evaluation of queries using operators of selection, projection, join and aggregation. Such systems combine traditional database algorithms and hashing functions.

Peer-to-Peer Infrastructure for Information Exchange and Retrieval (PIER) relies supposes that underlying layers (storage and lookup services) include special functionalities. In particular, they must notify data migration caused by configuration modifications (join/leave of peers), and more over they must allow suppression of data and group communication.

This assumption allows PIER to offer new functionalities. For example, the management of temporary resources, when data are stored only for a predefined time duration, can be used in complex query processing.

PIER evaluates SQL queries on tuples stored in the P2P system. It defines 14 logical operators, 26 physical operators and three types of index to evaluate queries. A set of optimized algorithms based on *symmetric hash join* is proposed and query evaluation may include re-hashing and/or temporary storage of intermediate results.

PIER allows the use of inequality terms and several indexes are proposed to enhance the query processing. *Prefix Hash Tree* [61] (PHT) is an example of one of them. Underlying DHT systems must provide the group concept, representing relation concept of the relational model, and temporary storage of data.

Framework for complex query processing (FCQ) is a framework which offers evaluation of relational operators like selection, projection, join and group by. Tuples of relational tables are stored and indexed in the underlying distributed storage service. The architecture of the system is based on a hierarchy of peers. A super-peer layer named *range guards* is used to store copies of tuples, a simple peer only stores pointers to super-peers.

FCQ uses a single index structure for the evaluation of all types of queries. It imposes special conditions on underlying systems and on hashing function as it uses a super peer layer and an order preserving hashing function. As a consequence, the evaluation of inequality terms is simple. FCQ, like MAAN, uses the successor function of the distributed storage service for query propagation. This has the same drawbacks as the ones already mentioned for MAAN.

3.4 Semi-structured or navigational model

This section deals with works on XML. A lot of recent proposals concern this kind of data. Among them DIP, KadoP and LDS, presented in the following. Query processing optimization, for example, by using bloom filters [38, 3] have been proposed so as some hints to reduce the size of indexing structures [13, 24].

Data Indexing in Peer-to-Peer DHT Networks (DIP) Data are shared in the DHT system and are described through the use of a descriptor. This descriptor is semi-structured XML data and has the form of XPath query. This descriptor is used for the generation of a set of so called "interesting queries" that index the data.

The query language is a subset of XPath queries which are "complete paths" in the XPath terminology [35].

As optimization DIP proposes the creation of indexed queries and the use of caches (see Section 4). These optimizations allow respectively an iterative evaluation of queries and the enhancement of the response time. Nevertheless, the mechanism to choose queries to be materialized may affect the performance and the coherency of the solution. DIP may be used on top of most DHT as it only uses simple functions (`put` and `get`).

KadoP [3, 4], has been built over FreePastry[4] [65] and uses Active XML [1]. It focuses on sharing XML, HTML or PDF documents as well as web services. These resources are described by using DTD files (conform to XML schemas [35]) or WSDL descriptions [78] for web services. Meta-data are structured using relations such as *partOf* and *isA*. Meta-data contain semantic concept that enrich queries on data. Concepts and values are linked through the use of the *relatedTo* relation. KadoP also uses *namespaces* to structure spaces of definitions. Meta-data are shared and stored in the P2P system. So data are indexed/published but original data are stored in the owning peer and are not stored in the DHT.

[4] A free implementation of Pastry.

Published data are identified by URIs or URLs which give localization information.

Different kinds of indexes are introduced. These indexes, registered in the DHT, allow the publication of name spaces, concepts, relations between concepts, sets of type in a namespace and finally the semantic link between two concepts.

All these allow the evaluation of semantically rich queries. Indexing and evaluation are closely related to LDS as shown in the next paragraph.

The main drawback of this solution is the fact that the quantity of information used for indexing concepts, relations between concepts is huge and requires a very large amount of storage resources. Techniques for going through this point may be found in [3].

Locating Data Sources in Large Distributed Systems (LDS) is an indexation service for XML data, it uses XPath for interrogation. LDS relies on traditional functions, so it is independent of the underlying P2P layers. Data stay in the owning peer and are published and described using "resumes". The first element of a resume is a set of paths to an attribute, the second is the set of values for this attribute and the third is the set of id of peers containing data. Resumes may be compressed using techniques like histogram [52] and bloom filters [43]. The scaling of this solution may be affected by the frequent use of specific meta-data. The peer in charge of the storage of this meta-data may overload. Therefore, a fragmentation strategy to deal with this issue is proposed. LDS also proposes the duplication of resumes linked to frequently used meta-data. But this solution is kept as a last recourse as problems of coherency may arise.

Proposed query processing reduces the volume of transfered data as intermediate answers do not contain all objects satisfying a query but only id of peers that are storing such object. As drawbacks, this introduces an additional step of evaluation to identify relevant objects in these peers. Proposed indexation may allow the evaluation of all types of queries in an homogeneous way, but indexes contain a huge quantity of data.

XPath for P2P (XP2P) presented in [13] deals with these problems and proposes to index only fragments of XML documents. In addition LDS indexes are affected by the arrival or departure of peers and this may be a problem in very dynamic systems.

3.5 Information retrieval trends

Infrastructure to support exact query evaluation is an important step forward, but important challenges still remain. One of them is dealing with semantic heterogeneity coming from a large number of different communities bridged together in a P2P system.

[81] deals with rich information retrieval in this context. This system doesn't try to offer exact answers. Instead, it seeks approximate answers like in retrieval information systems. Users are looking to find files "semantically close" to a file

they already have. The proposed solution is based on vectors of terms for indexation and querying, and on an order preserving hashing function for evaluation. Classical rappel and precision are used to measure the quality of an answer.

SPRITE focuses on reducing the amount of storage space required to index documents. The idea is to ignore the terms used for indexing documents that are never used in queries.

An other important trend concerns the huge number of shared objects and the potentially too large size of answers returned to a simple query. To deal with this problem DHTop [8] focuses on evaluating queries with a top-k [51] operator. Like PIER, DHTop deals with relational data. Special indexes are introduce to deal with inequality and top-k answers.

Based on the fact that the top-k operator is important in large centralized databases, it seems obvious that its study in P2P DHT systems is very important.

4 Query processing optimization

Sections 2 and 3 provide core functionalities enabling P2P DHT systems to be used in applications like web applications – search engines [55], P2P streaming [56]– as well as in the security domain– generation of digital certificates and intrusion detection [45]. In this style of applications, query processing optimization is essential. On the one hand, the systems in Section 2 improve maintenance processes and storage management for reducing the complexity of routing processes and resource usage. On the other hand, systems in Section 3, for enhancing query language capabilities, provide several kinds of evaluation strategies taking into account query style and the information about system state.

The use of optimizers for improving query performance is a typical practice in data management systems. Those optimizers use statistical information to support their decisions on the availability of structures as indexes and caches. This Section focuses on elements for supporting query processing optimization and provides a description in a top-down way according to abstraction levels. Section 4.1 presents optimizers based on static optimization, Section 4.2 analyzes elements such as distributed cache and Section 4.3 is about statistics.

4.1 Optimizers

P2P systems optimizers research faces several challenges. Major challenges are related to the impossibility of having global catalogs, the large number of peers and their dynamicity, and the difficulty to build/maintain statistics in these systems. The first part of this Section gives a brief description about some works on P2P optimizers, and the second part highlights decisions about optimization in some of the works described in Section 3.

P2P DHT Optimizers. Several works have analyzed and proposed different styles of P2P optimizers ranging from classic optimizers to optimizers based on user descriptions.

Classic optimizers. [14] proposes a P2P optimizer based on the super-peers existence and the concept of classic optimizers using distributed knowledge about schemes in the network as well as distributed indexes. This work proposes several execution plans considered as optimal and characterized by costs calculated using statistical information. Indexes provide information to super-peers for them to decide the place to execute a sub query. Two places are possible: locally in the contacted super-peer or in an other super-peer. The cost associated to one plan enables the selection of the best execution plan. This work does not consider the dynamicity of peers.

Optimization defined by the user. Correlated Query Process (CQP) [17] and PIER [32] are opposite proposals to the first one. In these cases, the user has all the responsibility of the query optimization process. CQP defines a template with information about peers to contact and about flows of data and processes. A query execution is made using that template for generating parallel execution between steps and using messaging for synchronization. On the other hand, PIER provides a language to define in a specific manner, a physical execution plan. This language named UFL (Unnamed Flow Language) contains logics and physical operators used by an expert user to submit a query. These operators do not include information about a peer responsible for its execution. This decision is taken for PIER according to index information. In these proposals the user is the key for providing a query optimization. PIER proposes a user interface to support physical plans building. Still, the assumption about user responsibility is very strong for inexpert users.

Range and join queries optimization. This Section presents some conclusions about optimization features included in the works of Section 3. Works will be presented according to the query type to be optimized.

Range queries. An interesting conclusion, based on MAAN and Mercury works, about range queries optimization is that term selectivity is important for deciding the terms execution order. In fact, terms execution order can impact the number of peers contacted. MAAN concludes that a term selectivity greater than 20% increases the number of contacted peers (that could be 50% of total peers) and the number of messages used in the strategy increases as well and could be close to the number of messages used by a flooding strategy. In the same way, Mercury shows that selection of the term with the lowest selectivity reduces by 25% or 30% the number of contacted peers (and the number of messages) compared to a random term selection.

PierSearch [47] (c.f. Section 4.2) explores the possibility to work with a DHT system and a non structured P2P system for obtaining a more complete answer as well as for improving the execution time. It shows that the answer recall increases according to a threshold used to identify rare objects. In particular, when the number of objects in a query result is 3 the answer recall is 95% and with 10 objects the answer recall is 99%. In other cases, queries can be executed in a more efficient manner using a non structured P2P system.

Join queries. Works as PIER and PinS highlight interesting results about the use of network resources in queries containing joins terms. PIER [33] proposes four join algorithms: Symmetric Hash Join (SHJ), Fetch Matches Join (FMJ), Symmetric Hash Semi Join and Bloom Filter - Symmetric Hash Join. It concludes that SHJ is the most expensive algorithm according to network resources, FMJ consumes an average of 20% and optimizations of SHJ use less resources, although this depends on the selectivity of conditions on the smaller relation. PinS [60] complements PIER work proposing three join query evaluation strategies: Index Based Join (IBJ), Nested Loop Join (NLJ) and NLJ with reduction in the search space. PinS concludes that IBJ needs, in most cases, less messages. When the join cardinality is less than the size of the search space NLJ is more efficient.

4.2 Cache

The literature reports specific proposals on caching related to P2P DHT systems [36, 47, 72, 10] but also adaptable and context aware caching services [23, 41, 53] that can be used in such systems. This Section presents first cache solutions implemented in P2P DHT Systems and then some proposals where DHT systems themselves are used as a cache.

DHT Systems using cache. At the distributed storage service level there are works as DHash [16] and PAST [67] (see Section 2) that propose a cache to ameliorate queries on popular objects. In those works the load distribution on peers is improved to reduce answer times. In the distributed data management service layer, there are works as DCT [72], PinS [75, 59] and CRQ [68] that improve the answer time by reducing the processing cost of queries previously executed. These proposals improve the traffic consumption and enable the use of a semantic cache for queries composed of equalities and inequalities terms. Although there are cache proposals in both distributed storage service and distributed data management service layers, it is difficult to reuse these solutions in new proposals as it is not clear how to provide cooperation between heterogeneous caches and as they do not give information about the context in which they are used. In the next part of this Section, there is a brief description of some representative works.

PAST and DHash are quite similar in their behavior. They propose popular objects for caching using space available on peers. PAST takes advantage of the peers contacted during a routing process to cache objects. An object in the PAST cache is a couple composed of a key and a set of identifier related to this key. A peer contacted in a routing process searches locally in its cache one entry to answer the request. Then, when an cache entry is not found the typical routing process continues. The cache entries are evicted from the cache using a GD-S [67] policy and are discarded at any time.

DIP (see Section 3) uses a query cache and entries in the cache contain the query with its identifier and all descriptors associated to objects shared in the system. These entries are registered in the peer contacted to evaluate the submitted query or in all peers visited for providing the query evaluation. In a search process, the local cache is visited before going to the peer responsible for the submitting query identifier. DIP proposes LRU as replacement policy.

CRQ provides a cache for range queries using CAN [63] as a distributed lookup service. CRQ has two types of exclusive entries. The first one is entries about row identifiers of specific query. The second one contains information about peers storing in their cache answers about a query. Once a query is required to a peer, it searches in peers associated with the complete query or to a sub query. Consequently, the complete answer for the submitted query or a partial answer, used to calculate the total answer, can be found in the distributed cache. CRQ guaranties a strong coherence, updating the cache when new objects are inserted in the system. Moreover, CRQ proposes the use of LRU as a replacement policy.

PinS proposes two query caching solutions. The first one [75] about queries including only equality terms whereas in the second case, PinS proposes a cache including range queries. Unlike other cache proposals, in the first cache, PinS provides a prefetch cache using statistics about queries frequency to identify the most popular queries. These queries are the candidates to be stored in the cache. As a new object arrives in the system, all entries containing queries that include the new object are updated, providing a strong coherence. A cache entry is indexed by a term identifier and contains a query with all objects shared in the system verifying its conditions. This tuple is stored in the peer responsible for the term identifier, related to the first term, in alphabetic order, of a query. When an equality query is demanded, PinS contacts all peers responsible for terms included in the query. These peers search in their local caches, queries related to the term demanded. When a peer finds a query closest to the query required by the user, it sends as answer the cache entry and not the answer related to the term that it manages.

The second cache type [59] includes range terms evaluation. In this case, a cache entry is indexed by the query identifier. The cache entry contains information about a query and a couple $atr_a = val_j$ together with objects identifiers associated to the attribute value. Queries are normalized to find the couple representative of the query.

The cache entries enable to evaluate exact and partial coincidence of queries within cache. Each entry uses a TTL to determine their freshness as well as a Time to Update. Both PinS proposals use LRFU as replacement policy.

DCT identifies frequent queries as candidates to be stored in the cache. These queries are stored as entries in a local peer only if there is available storage space. Otherwise, the results set of these queries is shortened and top k answers are selected and stored in the cache. Finally, if the space is not enough, a replacement policy is used to identify the entry cache to be deleted from the cache. There

are two kinds of entry cache. The first one is indexed by a term contained in the requested query. This term is selected in a random manner. This entry contains all or partial document identifiers that answer the submitted query. The second one is indexed according to the document identifier (URI) and contains a frequent query together with all URI's satisfying the query. When a query is subsumed in the system, this query is decomposed in sub queries to identify peers with information into the cache to answer the query. When cache information is not enough to answer a query, a broadcast to all peers responsible for one of the query terms is made.

DHT Systems used as cache. Squirrel [36] and PierSearch [47] are examples of DHT systems used as cache. Squirrel tries to reduce the use of network resources and PierSearch improves the answer recall. It is difficult to generalize a cache proposal because they are coupled with the core proposals and there are some important issues like object placement, replacement policy and coherence models, that are not described or studied. The rest of this Section presents a brief description about some representative works.

Squirrel enhances searches on web pages. In the same way as CRQ, it proposes two storage techniques. Both of them use the URL of a web page for generating object identifiers. In the first technique, a cache entry is indexed by URL's and contains the web page identified by the URL. In the second one, a cache entry is composed of information distributed in several peers. The peers demanding a query on a WebPage, named delegate peers, store the first type cache entries. Additionally, the peer (P_r) responsible for the URL key stores information about delegate peers maintained in their caches information about the web page involved in the query. When a web page is demanded, the browser searches the page in its local cache. When the page is not found or the version stored is not adequate, Squirrel is contacted to search the page. Squirrel searches in P_r the web page or information about delegate peers. Finally, when a miss is produced, Squirrel searches the web page in the server. Squirrel uses TTL as freshness technique and LRU for replacement policy.

PierSearch improves query evaluation on objects, considered as rare for Gnutella, on a P2P DHT System. Several strategies are studied to classify an object as rare. For example, an object is considered rare if the use frequency of one meta-data (associated to this object) or the number of answer query (including this object), are lower than a threshold. Two types of entry cache are proposed. The first one, indexed by all object information like *name, size, IPAdress, and PortNumber*, contains all this information. The second one, is indexed by one meta-data, and contains all meta-data associated to one object and its object identifier. The use of PierSearch occurs when a rare object is requested in Gnutella. In this case, the search is made using PierSearch and not the typical search on Gnutella. TTL are used as in Squirrel.

4.3 Statistics

Statistics enable optimizers to decide on the best execution plan including decisions about peers to contact, physic algorithms to use as well as creation of new indexes or the inclusion of new terms in an index. In a general context, a statistic is information about a system including a storage value, and a time to live. Statistics management is a big difficulty one faces. Consequently, a process to update statistic values and to identify events affecting the statistic will be provided The use of notification mechanisms as publication/subscription are a typical practice for communicating events to peers involved in statistic management.

Several works tackle issues about statistics in P2P. Still, the number of works is low with regards to works about declarative queries on P2P DHT. The main topics analyzed in statistics research concern decisions about what statistics will be gathered and how to provide statistics management. This Section focuses on the second topic because it is closest to declarative queries and presents some works with different characteristics to exemplify the research about statistics on P2P context.

The statistics works in P2P DHT systems can be evaluated according to distinct quality attributes [54]. The most important attributes are efficiency, scalability, load balance and accuracy. Efficiency is related to the number of peers contacted during the calculation of the statistic value. Scalability depends on the way to distribute the calculation process; load balancing is focused on increasing robustness in the statistic management process according to storage and access policies; and accuracy affects the quality of decision process using statistics values.

Mercury [11] provides an evaluation of queries composed of several attributes, as described in Section 3. Mercury uses statistics to provide a load balance according to storage dimension as well as to estimate selectivity of queries. It gathers information about data distribution and number of peers classified by range values. Additionally, it provides random sampling as statistic management strategy to maintenance local histograms.

PISCES [79] provides data management based on BATON (see Section 2). Unlike other works, PISCES proposes a mechanism to identify in a dynamic manner terms to be indexed in the system. The use of statistics enables dynamic indexation. In fact, PISCES gathers information about the number of peers in the system, average churn rates, number of all executed queries, and query distribution. PISCES uses a gossip protocol [54] for providing maintenance on local histograms.

Smanest [58] is a P2P service, contrary to the first's proposals, for providing statistics management on top of P2P DHT systems. It enables creation, maintenance and deletion of different kinds of statistics in a decoupling manner of the P2P DHT system. At the same time, Smanest provides a uniform access

to statistics as well as guarantees a transparent solution according to statistics placement. An update process is proposed, giving to the user the option to determine the kind of update policy. Two types of updated policies are provided: immediate and deferred. Immediate specifies the update on statistic value after an event changes it. Whereas, deferred policy defines a time period or a number of events to trigger the update process. In the last two cases, the statistical accuracy can be affected. Smanest proposes an automatic mechanism for statistics management, based on events monitoring, and for events notification in an efficient manner.

In fact, collecting statistics in P2P systems is a crucial and hard task. Crucial because they will be needed to allow good performances and hard because the usual way to deal with them are not well fitted to P2P systems. The same kind of challenges appear for privacy concerns which are the focus of the next Section.

5 Data privacy

This section analyses privacy issues addressed by P2P DHT systems. It first introduces the data privacy problem and then discusses access restriction (Section 5.2), anonymity (Section 5.3) and trust techniques (Section 5.4).

5.1 Rationale

Currently, the democratization of the use of information systems and the massive data digitalization allow us to identify all aspects of the a person's life. For instance, their professional performance (e.g., publish or perish software, dblp website), their client's profile (e.g., thanks to fidelity smart cards), their user's profile (e.g., thanks to their user identity, access localities, generated traffic) or their health level (e.g., thanks to the digitalization of medical records). Those issues have given rise serious data privacy concerns.

P2P data sharing applications, due to their open and autonomous nature, highly compromise the privacy of data and peers. The P2P environment can be considered as hostile because peers are potentially untrustworthy. Data can be accessed by everyone, used for everything (e.g., profiling, illegal competition, cheating, marketing or simply for activities against the owner's preferences or ethics) and a peer's behavior and identity can be easily revealed. It is therefore necessary, in addition to other security measures, to ensure peer anonymity, improve data access control and use/adapt trust techniques in P2P systems.

To limit the pervasiveness of privacy intrusion, efforts are being made in data management systems [57]. Concerning P2P systems, several proposals take into account security issues (integrity, availability and secrecy) that are closely related to privacy. For instance, in [9], authors discuss various P2P content distribution approaches addressing secure storage, secure routing and access controle; [71] proposes an analysis of security issues in P2P DHT systems. It provides a description of security considerations when peers in the DHT system do not follow the protocol correctly, particularly secure assignment of node IDs, secure

maintenance of routing tables, and secure forward of messages; and [48] looks into the security vulnerabilities of overlay networks.

Nevertheless, very few works propose solutions to preserve privacy. In [12] a comparison of several unstructured P2P systems is made. The comparison focuses on four criteria, namely, uploader/downloader anonymity, linkability (correlation between uploaders and donwloaders) and content deniability (possibility of denying the knowledge of the content transmitted/stored). [9] also presents an analysis of authentication and identity management. It is worth noting that most of these works concern unstructured P2P systems which are outside of the scope of this paper.

This section discusses the three data sharing P2P DHT systems that, to our knowledge, address data privacy in P2P DHT systems – PAST, OceanStore and PriServ.

PAST [67], as mentioned in Section 2 is a distributed storage utility located in the storage management layer (see Figure 1). Besides providing persistence, high availability, scalability and load balancing it focuses on security issues. In PAST, peers are not trusted (except requester peers) and its proposal limits the potentially negative impact of malicious peers.

OceanStore [44] is a utility infrastructure designed for global scale persistent storage which relies on Tapestry [80]. It was designed to provide security and high data availability. As in PAST, server peers are not trusted.

PriServ [39, 40] is the privacy service of the APPA infrastructure [7]. It was designed to prevent malicious data access by untrusted requesters. APPA has a network-independent architecture that can be implemented over various structured and super-peer P2P networks. The PriServ prototype uses Chord [73] as overlay network (similar to KSS, MAAN and FCQ (see Section 2)).

OceanStore and PriServ provide distributed data management services (see Figure 1). They were not analyzed in the preceding sections of this paper because they do not propose particular querying mechanisms. They use a basic hash key search. However, OceanStore and PriServ propose interesting solutions to enforce data privacy.

The next sections analyze the three aforementioned systems with respect to access restrictions, anonymity support and trust management.

5.2 Access restriction

In OceanStore, non public data is encrypted and access control is based on two types of restrictions: *reader* and *writer* restrictions. In the *reader* restriction, to prevent unauthorized reads, data are encrypted (with symmetric-keys). Encryption keys are distributed to users with read permissions. To revoke the read permission, the data owner requests that the replicas be deleted or re-encrypted with a new key. A malicious reader is able to read old data from cached copies or from misbehaving servers that fail to delete or re-key. This problem is not

specific to OceanStore, even in conventional systems there is no way to force a reader to forget what has been read.

To prevent unauthorized writes, writes must be signed so that well-behaved servers and clients can verify them against an access control list (ACL). The owner of an object can choose the ACL for an object by providing a signed certificate. ACL are publicly readable so that server peers can check whether a write is allowed. Thus, servers restrict writes by ignoring unauthorized updates.

In PriServ, the access control approach is based on Hippocratic databases [6] where *access purposes* are defined to restrain data access. Additionally, in PriServ authors propose to take into account the operation (read, write, disclosure) that will be applied. The idea is that in order to obtain data, peers specify the purpose and the operation of the data request. This explicit request commits clients to use data only for specified purposes and operations. Legally, this commitment, may be used against malicious clients if data is used for other purposes/operations.

To make data access control, purposes and operations are included in the generation of data keys. For this a publicly known hash function hashes the data reference, the access purpose and the operation (it is considered that those parameters are known by peers). Thus, the same data with different access purposes and different operations have different keys. According to the authors, previous studies have shown that considering 10 purposes allows to cover a large number of applications. In addition, the number of operations considered is only 3.

Server peers are untrusted in PriServ. Two functions to distribute data are proposed: *publishReference()* and *publishData()*. In the first one, owner and data references are distributed in the system but not data themselves. When a peer asks for data, server peers return only the data owner reference. Requester peers must contact data owners to obtained data. In the second function, encrypted data is distributed. When a peer requests data, server peers return the encrypted data and the data owner reference that stores the decryption key. Owner peers use public-key cryptography to send decryption keys to requester peers.

In PriServ a double access control is made during data requesting. Servers make an access control based on the information received during data distribution. A more sophisticated access control is made by owners where, in particular, trust levels of requester peers are verified. Malicious acts by servers are limited because, for each request, peers contact data owners to obtain data (if data have been published with the publishReference() function) or encryption keys (if the publishData() function has been used.

In several systems, the lack of authentication is overcome by the distribution of the encryption keys, necessary for accessing content, to a subset of privileged users. As in OceanStore and PriServ, in PAST, users may encrypt their data before publishing. This feature conduces to the *deniability of stored content* because nodes storing or routing encrypted data cannot know their content. This protects router or server privacy because they are not responsible for the content they transfer/store.

5.3 Anonymity

In [22], the authors underline the need for anonymity in P2P systems. Anonymity can enable censorship resistance, freedom of speech without the fear of persecution, and privacy protection. They define four types of anonymity: 1) author anonymity (which users created which documents?), 2) server anonymity (which nodes store a given document?), 3) reader anonymity (which users access which documents?) and 4) document anonymity (which documents are stored at a given node?).

In PAST, each user holds an initially unlinkable pseudonym in the form of a public key. The pseudonym is not easily linked to the user's actual identity. If desired, a user may have several pseudonyms to obscure that certain operations were initiated by the same user. PAST users do not need to reveal their identity, the files they are retrieving, inserting or storing.

PriServ and OceanStore do not use anonymity techniques. In systems where anonymity is not taken into account, it is possible to know which peer potentially can store which data. This knowledge facilitates data censorship [25, 31, 69].

5.4 Trust techniques

[50] discuses the complex problem of trust and reputation mechanisms and also analyzes several distributed systems and some unstructured overlay systems.

In PAST, requester peers trust owner et server peers thanks to a smartcard held by each node which wants to publish data in the system. A private/public key pair is associated with each card. Each smartcard's public key is signed with the smartcard issuer's private key for certification purposes. The smartcards generate and verify various certificates used during insert and reclaim operations and they maintain secure storage quota system. A smartcard provides the node ID for an associated PAST node. The node ID is based on a cryptographic hash of the smartcard's public key. The smartcard of a user wishing to insert a file into PAST issues a file certificate. The certificate contains a cryptographic hash of the file's contents (computed by the requested node) and the fileId (computed by the smartcard) among others.

In PriServ, every peer has a trust level. An initial trust level is defined depending on the *quality* of peers, then it evolves depending on the peer's reputation in the system. There is no a global trust level for a peer. The perception of the trustworthiness of peer A may be different for that of peer B or even that of the peer C. Locally, peers have a trust table which contains the trust level of some peers in the system.

If a server peer does not know the trust level of the requesting peer locally, it asks its *friends* (peers having a high trust level in its trust table), and if a friend does not have the requested trust level, it asks for it from its friends like a flooding limited by a TTL (time to live). If a peer does not have friends it will request the trust level from all peers in its finger table (log(N) peers).

6 Conclusion and perspectives

P2P DHT systems clearly provide a powerful infrastructure for massively distributed data sharing. This paper introduced first a global functional architecture which distinguishes the underlying distributed lookup services to maintain the P2P overlay network, then a distributed storage service layer providing data persistency (involving data migration, replication and caching), and then a last layer consisting of high level data management services as declarative query processors. The paper then extensively discussed query and data privacy supports. These discussions reveal some maturity in querying solutions but also deficiencies on providing data privacy.

This paper considered many proposals on query processing on top of P2P DHT systems covering a large variety of data models and query languages. Such proposals allow queries on shared data in a "P2P database" style as well as with less structured approaches. That is either by providing querying support on meta-data associated to potentially any object (whatever its type), or by using approximate queries in a "P2P information retrieval" style.

Optimization issues have been analyzed as they are crucial in developing efficient and scalable P2P query processors. As such query processors operate in dynamic massively distributed systems without global control and complete statistics, query optimization becomes very hard. For almost each operator of the query language it is necessary to find a specific optimization solution. Join operators, range queries and top-k queries have been given particular attention because their evaluation can be extremely time and resource consuming. Caching issues have also been presented, mainly as an approach to improving the data management services. Several operational solutions exist but stale data management is not yet optimum.

The last topic discussed in, this paper was data privacy support. This major issue has surprisingly received little attention until now in P2P DHT systems (most efforts concern unstructured P2P systems). Some access restriction techniques, anonymity support and trust management have been proposed but more complete proposals are required.

Providing appropriate privacy is certainly essential to allow more applications (particularly industrial ones) to rely on P2P data sharing. Thus this important issue has several interesting perspectives and challenging open problems. For example, more effort is needed to prevent and limit data privacy intrusion but also to verify that data privacy has been preserved. Verification can be made with auditing mechanisms that can be based on techniques like secure logging [18] of data access or even watermarking [5]. An auditing system should detect violations of privacy preferences and punishments (or rewards) to misbehaving (or honest) peers should be considered. It is important to notice the existence of a tradeoff between anonymity and data access control. Veri?cation can hardly be implemented in environments where everyone is anonymous.

Other privacy issues concern the distributed storage services. For instance how to avoid storing data on peers that are untrustworthy by data owners. Currently, data are stored on live peers whose key is the closest one to the data

key (e.g., the successor function of Chord). It could be interesting to define DHT where data owners can influence the distribution of their data.

Further important research perspectives concern data consistency and system performances. Data consistency issues are out of the scope of this paper but are nevertheless important in data sharing systems. Data shared through P2P DHT system have been considered as read only by the majority of proposals. One of the main reasons is the impossibility to guarantee update propagation to all peers holding a copy of data[5]. The literature reports recent proposals on data consistency support in P2P DHT systems but this aspect needs further research.

System performance also reveals some topics for research. One problem is cross layer optimization combining the optimization solutions implemented by the distributed storage services and data management services. For example, such services may propose various caching approaches that should be "coordinated" to tune the system. Another important aspect is better support of peers volatility and dynamic system configuration by the data management services. Current query supports implicitly consider P2P systems with low churn rate. Such supports could become inefficient or not work when there is a high churn rate in the system. Data management services should therefore be highly adaptable and context aware. This will be even more important if mobile devices participate in the system.

Lastly, the issue of semantic heterogeneity of shared data is important in all data sharing systems and is even more accentuated in P2P environments where peers are extremely autonomous.

There are certainly other important unsolved problems but the aforementioned ones already will lead to much exciting research.

Acknowledgement: Many thanks to Solveig Albrand for her help with this paper.

References

1. S. Abiteboul, O. Benjelloun, I. Manolescu, T. Milo, and R. Weber. Active XML: A Data-Centric Perspective on Web Services. In *Demo Proc. of Int. Conf. on Very Large Databases (VLDB)*, Hong Kong, China, August 2002.
2. S. Abiteboul, I. Dar, R. Pop, G. Vasile, and D. Vodislav. EDOS Distribution System: a P2P Architecture for Open-Source Content Dissemination. In *IFIP Working Group on Open Source Software (OSS)*, Limerick, Ireland, June 2007.
3. S. Abiteboul, I. Manolescu, N. Polyzotis, N. Preda, and C. Sun. XML Processing in DHT Networks. *Int. Conf. on Data Engineering Data Engineering (ICDE)*, April 2008.
4. S. Abiteboul, I. Manolescu, and N. Preda. Sharing Content in Structured P2P Networks. In *Journées Bases de Données Avancées*, Saint-Malo, France, October 2005.

[5] disconnected peers could conserve stale copies.

5. R. Agrawal, P. Haas, and J. Kiernan. A System for Watermarking Relational Databases. In *Int. Conf. on Management of Data (SIGMOD)*, San Diego, California, USA, June 2003.
6. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *Int. Conf. on Very Large Databases (VLDB)*, Hong Kong, China, August 2002.
7. R. Akbarinia, V. Martins, E. Pacitti, and P. Valduriez. Design and Implementation of APPA. *Global Data Management (Eds. R. Baldoni, G. Cortese, F. Davide), IOS Press*, 2006.
8. R. Akbarinia, E. Pacitti, and P. Valduriez. Processing Top-k Queries in Distributed Hash Tables. In *Euro-Par 2007 Parallel Processing*, Rennes ,France, August 2007.
9. S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4), 2004.
10. M. S. Artigas, P. G. López, and A. F. Gómez-Skarmeta. Subrange Caching: Handling Popular Range Queries in DHTs. In *Int. Conf. on Data Management in Grid and Peer-to-Peer Systems (Globe)*, 2008.
11. A. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. In *Int. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, Portland, Oregon, USA, August-September 2004.
12. R. Blanco, N. Ahmed, D. H. L. Sung, H. Li, and M. Soliman. A Survey of Data Management in Peer-to-Peer Systems. Technical Report CS-2006-18, University of Waterloo, 2006.
13. A. Bonifati and A. Cuzzocrea. Storing and Retrieving XPath Fragments in Structured P2P Networks. *Data & Knowledge Engineering*, 59(2), 2006.
14. I. Brunkhorst, H. Dhraief, A. K. A, W. Nejdl, and C. Wiesner. Distributed Queries and Query Optimization in Schema-Based P2P-Systems. In *Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, Berlin, Germany, September 2003.
15. M. Cai, M. Frank, J. Chen, and P. Szekely. MAAN: A Multi-Attribute Addressable Network for Grid Information Services. In *Int. Workshop on Grid Computing (GRID)*, Phoenix, Arizona, November 2003.
16. J. Cates. *Robust and Efficient Data Management for a Distributed Hash Table.* Master thesis, Massachusetts Institute of Technology, USA, May 2003.
17. Q. Chen and M. Hsu. Correlated Query Process and P2P Execution. In *Int. Conf. on Data Management in Grid and Peer-to-Peer Systems (Globe)*, Turin, Italy, September 2008.
18. C. N. Chong, Z. Peng, and P. H. Hartel. Secure Audit Logging with Tamper-Resistant Hardware. In *Int. Conf. on Information Security (SEC)*, Athens, Greece, May 2003.
19. G. D. Costa, S. Orlando, and M. D. Dikaiakos. Multi-set DHT for Range Queries on Dynamic Data for Grid Information Service. In *Int. Conf. on Data Management in Grid and Peer-to-Peer Systems (Globe)*, 2008.
20. F. Dabek, M. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area Cooperative Storage with CFS. In *Int. Sysposium on Operating Systems Principles (SOSP)*, Banff, Canada, October 2001.
21. F. Dabek, B. Y. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a Common API for Structured Peer-to-Peer Overlays. In *Int. Workshop on Peer-to-Peer Systems (IPTPS)*, Berkeley, CA, USA, February 2003.
22. N. Daswani, H. Garcia-Molina, and B. Yang. Open Problems in Data-Sharing Peer-to-Peer Systems. In *Proc Int. Conf. on Database Theory*, Siena, Italy, January 2003.

23. L. d'Orazio, F. Jouanot, C. Labbé, and C. Roncancio. Building Adaptable Cache Services. In *Int. Workshop on Middleware for Grid Computing (MGC)*, Grenoble, France, November 2005.
24. F. Dragan, G. Gardarin, B. Nguyen, and L. Yeh. On Indexing Multidimensional Values in A P2P Architecture. In *French Conf. on Bases de Données Avancées (BDA)*, Lille, France, 2006.
25. R. Endsuleit and T. Mie. Censorship-Resistant and Anonymous P2P Filesharing. In *Int. Conf. on Availability, Reliability and Security (ARES)*, Vienna, Austria, April 2006.
26. P. Furtado. Schemas and Queries over P2P. In *Int. Conf. on Database and Expert Systems Applications (DEXA)*, 2005.
27. L. Galanis, Y. Wang, S. Jeffery, and D. DeWitt. Locating Data Sources in Large Distributed Systems. In *Int. Conf. on Very Large Databases (VLDB)*, Berlin, Germany, September 2003.
28. L. Garcés-Erice, P. Felber, E. Biersack, and G. Urvoy-Keller. Data Indexing in Peer-to-Peer DHT Networks. In *Int. Conf. on Distributed Computing Systems (ICDCS)*, Columbus, Ohio, USA, June 2004.
29. O. Gnawali. *A Keyword-Set Search System for Peer-to-Peer Networks.* Master thesis, Massachusetts Institute Of Technology, Massachusetts, USA, June 2002.
30. N. Harvey, M. Jones, S. Saroiu, M. Theimer, and A. Wolman. SkipNet: A Scalable Overlay Network with Practical Locality Properties. In *Int. Symposium on Internet Technologies and Systems (USITS)*, Washington, USA, March 2003.
31. S. Hazel, B. Wiley, and O. Wiley. Achord: A Variant of the Chord Lookup Service for Use in Censorship Resistant Peer-to-Peer Publishing Systems. In *Int. Workshop on Peer To Peer Systems (IPTPS)*, Cambridge, MA, USA, March 2002.
32. R. Huebsch. *PIER: Internet Scale P2P Query Processing with Distributed Hash Tables.* Phd thesis, EECS Department, University of California, Berkeley, California, USA, May 2008.
33. R. Huebsch, B. Chun, J. Hellerstein, B. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. Ymerefendi. The Architecture of PIER: An Internet-Scale Query Processor. In *Int. Conf. on Innovative Data Systems Research (CIDR)*, California, USA, January 2005.
34. R. Huebsch, J. Hellerstein, N. Lanham, B. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *Int. Conf. on Very Large Databases (VLDB)*, Berlin, Germany, September 2003.
35. D. Hunter. *Initiation XML.* Editions Eyrolles, 2001.
36. S. Iyer, A. Rowstron, and P. Drushchel. Squirrel - A Decentralized Peer-to-Peer Web Cache. In *Int. Symposium on Principles of Distributed Computing (PODC)*, California, USA, July 2002.
37. H. Jagadish, B. Ooi, and Q. Vu. Baton: A Balanced Tree Structure for Peer-to-Peer Networks. In *Int. Conf. on Very Large Databases (VLDB)*, Trondheim, Norway, September 2005.
38. C. Jamard, G. Gardarin, and L. Yeh. Indexing Textual XML in P2P Networks Using Distributed Bloom Filters. In *Int. Conf. on Database Systems for Advanced Applications (DASFAA)*, 2007.
39. M. Jawad, P. Serrano-Alvarado, and P. Valduriez. Design of PriServ, A Privacy Service for DHTs. In *Int. Workshop on Privacy and Anonymity in the Information Society (PAIS)*, Nantes, France, March 2008.
40. M. Jawad, P. Serrano-Alvarado, P. Valduriez, and S. Drapeau. Data Privacy in Structured P2P Systems with PriServ. Submitted paper, May 2009.

41. F. Jouanot, L. D'Orazio, and C. Roncancio. Context-Aware Cache Management in Grid Middleware. In *Int. Conf on Data Management in Grid and Peer-to-Peer Systems (Globe)*, Turin, Italy, September 2008.
42. D. D. Judd. Geocollaboration using Peer-Peer GIS, May 2005. http://www.directionsmag.com/article.php?article_id=850.
43. D. Kossmann. The State of the Art in Distributed Query Processing. *ACM Computing Surveys*, 32(4), 2000.
44. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W.Weimer, C.Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, November 2000.
45. F. Lesueur, L. Mé, and V. V. T. Tong. A Distributed Certification System for Structured P2P Networks. In *Int. Conf. on Autonomous Infrastructure, Management and Security (AIMS)*, Bremen, Germany, July 2008.
46. Y. Li, H. V. Jagadish, and K.-L. Tan. SPRITE: A Learning-Based Text Retrieval System in DHT Networks. In *Int. Conf. on Data Engineering (ICDE)*, 2007.
47. B. Loo, J. Hellerstein, R. Huebsch, S. Shenker, and I. Stoica. Enhancing P2P File-Sharing with an Internet-Scale Query Processor. In *Int. Conf. on Very Large Databases (VLDB)*, Toronto, Canada, August-September 2004.
48. E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys and Tutorials*, 7, 2005.
49. D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In *Int. Symposium on Principles of Distributed Computing (PODC)*, Monterey, CA, USA, July 2002.
50. S. Marti and H. Garcia-Molina. Taxonomy of Trust: Categorizing P2P Reputation Systems. *Computer Networks*, 50(4), 2006.
51. S. Michel. *Top-k Aggregation Queries in Large-Scale Distributed Systems*. Phd thesis, Saarland University, Saarbrucken, Germany, May 2007.
52. H. Molina, J. Ullman, and J. Widom. *Database System Implementation*. Prentice Hall, 2000.
53. A. Mondal, S. K. Madria, and M. Kitsuregawa. CLEAR: An Efficient Context and Location-Based Dynamic Replication Scheme for Mobile-P2P Networks. In *Int. Conf. on Database and Expert Systems Applications (DEXA)*, 2006.
54. N. Ntarmos, P. Triantafillou, and G. Weikum. Counting at Large: Efficient Cardinality Estimation in Internet-Scale Data Networks. In *Int. Conf. on Data Engineering (ICDE)*, Atlanta, USA, April 2006.
55. Open-Source Search Engine. YACY, 2009. http://yacy.net/.
56. P2P Streaming. Joost, 2009. http://www.joost.com/.
57. M. Petkovic and W. J. Eds. *Security, Privacy, and Trust in Modern Data Management*. Data-Centric Systems and Applications. Springer, 2007.
58. C. Prada. *Servicio para Manejar Estadísticas en Sistemas P2P Basados en DHT*. Master thesis, Universidad de los Andes, Bogota, Colombia, January 2009.
59. C. Prada, C. Roncancio, C. Labbée, and M.P.Villamil. Semantic Caching Proposal in a P2P Querying System. In *Congreso Latinoamericano de Computacin de Alto Rendimiento*, Santa Marta, Colombia, June 2007.
60. C. Prada, M. Villamil, and C. Roncancio. Join Queries in P2P DHT Systems. In *Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, Auckland, New Zealand, August 2008.

61. S. Ramabhadran, S. Ratnasamy, J. Hellerstein, and S. Shenker. Prefix Hash Trees An Indexing Data Structure Over Distributed Hash Tables, 2004. http://berkeley.intel-research.net/sylvia/pht.pdf.
62. A. Ramachandran and N. Feamster. Authenticated Out-of-Band Communication Over Social Links. In *Int. Workshop on Online social networks (WOSN)*, Seattle, WA, USA, August 2008.
63. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. In *Int. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, San Diego, CA, USA, August 2001.
64. P. Reynolds and A. Vahdat. Efficient Peer-to-Peer Keyword Searching. In *Int. Middleware Conf.*, Rio de Janeiro, Brasil, June 2003.
65. Rice University Houston, USA. FreePastry, 2002. http://freepastry.rice.edu/FreePastry/.
66. A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Int. Conf. on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, November 2001.
67. A. Rowstron and P. Druschel. Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-Peer Storage Utility. In *Int. Symposium on Operating Systems Principles (SOSP)*, Banff, Canada, October 2001.
68. O. Sahin, A. Gupta, D. Agrawal, and A. El-Abbadi. A Peer-to-Peer Framework for Caching Range Queries. In *Int. Conf. on Data Engineering (ICDE)*, Boston, USA, March-April 2004.
69. A. Serjantov. Anonymizing Censorship Resistant Systems. In *Int. Workshop on Peer To Peer Systems (IPTPS)*, Cambridge, MA, USA, March 2002.
70. S. Shing, G. Yang, D. Wang, J. Yu, S. Qu, and M. Chen. Making Peer-to-Peer Keyword Searching Feasible Using Multi-level Partitioning. In *Proc. Int. Workshop on Peer-to-Peer Systems (IPTPS)*, San Diego, CA, USA, February 2004.
71. E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Int. Workshop on Peer To Peer Systems (IPTPS02)*, Cambridge, MA, USA, March 2002.
72. G. Skobeltsyn and K. Aberer. Distributed Cache Table: Efficient Query-Driven Processing of Multi-Term Queries in P2P Networks. In *Int. Workshop on Information Retrieval in Peer-to-Peer Networks (P2PIR)*, Arlington, USA, November 2006.
73. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Int. Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, San Diego, CA, USA, August 2001.
74. P. Triantafillou and T. Pitoura. Toward a Unifying Framework for Complex Query Processing over Structured Peer-to-Peer Data Networks. In *Int. Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*, Berlin, Germany, September 2003.
75. M. Villamil. *Service de Localisation de Données pour les Systèmes P2P*. Phd thesis, Institut National Polytechnique de Grenoble, Grenoble, France, June 2006.
76. M. Villamil, C. Roncancio, and C. Labbé. PinS: Peer to Peer Interrogation and Indexing System. In *Int. Database Engineering and Applications Symposium (IDEAS)*, Coimbra, Portugal, June 2004.
77. M. Villamil, C. Roncancio, and C. Labbé. Querying in Massively Distributed Storage Systems. In *Journées Bases de Données Avancées*, Saint-Malo, France, October 2005.

78. WSDL. Web Services Description Language (WSDL) 1.1, 2001. http://www.w3.org/TR/wsdl.
79. S. Wu, J. Li, B. Ooi, and K.-L. Tan. Just-in-Time Query Retrieval over Partially Indexed Data on Structured P2P Overlays. In *Int. Conf. on Management of Data (SIGMOD)*, Vancouver, Canada, June 2008.
80. B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), 2004.
81. Y. Zhu and Y. Hu. Efficient Semantic Search on DHT Overlays. *Parallel and Distributed Computing*, 67(5), 2007.